

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patent Application

5 Applicant(s): DePauw et al.
Docket No.: YOR920010309US2
Serial No.: 10/040,344
Filing Date: January 2, 2002
Group: 2193
10 Examiner: Jason D. Mitchell

Title: Method and Apparatus for Tracing Details of a Program Task

15

CORRECTED APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
20 P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

25 Appellants hereby submit this Corrected Appeal Brief to conform to the current format requirements. The original Appeal Brief was submitted on October 7, 2005 to appeal the final rejection dated May 5, 2005, of claims 1 through 35 of the above-identified patent application.

30

REAL PARTY IN INTEREST

The present application is assigned to International Business Machines Corporation, as evidenced by an assignment recorded on January 2, 2002 in the United States Patent and Trademark Office at Reel 012466, Frame 0343. The assignee, International Business Machines Corporation, is the real party in interest.

35

RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

STATUS OF CLAIMS

Claims 1 through 35 are pending in the above-identified patent application. Claims 1-2, 17-18, 20-23, and 28-30 remain rejected under 35 U.S.C. §101 because the claimed invention is directed to non-statutory subject matter. Claims 1-35 5 remain rejected under 35 U.S.C. §102(b) as being anticipated by Laffra et al. (United States Patent Number 5,832,270). Claims 1-4, 6-8, 17, 18, 20-25, 28-30, and 32-35 are being appealed.

STATUS OF AMENDMENTS

10 There have been no amendments filed subsequent to the final rejection.

SUMMARY OF CLAIMED SUBJECT MATTER

The present invention is directed to a method and apparatus for analyzing one or more program tasks associated with a software system (110, 120). A program 15 task-oriented tracing and analysis technique allows detailed information to be gathered and analyzed for one or more specified program tasks. (Page 3, line 22, to page 4, line 19.) A user can iteratively vary the level of detail of the selected program task(s) of interest, or both, until the source of a problem is identified. For each program task (105) under analysis, the user can define what commences a task and what concludes a task. 20 (Page 4, line 20, to page 5, line 26.) A software program is monitored until the user-specified criteria for commencing a task is identified and continues to trace the execution of the software program until the user-specified criteria for concluding a task is identified. (Page 6, line 1, to page 9, line 7.)

In one exemplary embodiment, a method (300) for analyzing behavior of a 25 software system is disclosed (page 3, line 22, to page 4, line 19), comprising: collecting details associated with a program task associated with said software system based on a specification associated with said program task, wherein said specification contains one or more conditions to initiate a trace of said program task (page 4, line 20, to page 9, line 7); and providing said collected details for analysis (page 3, line 22, to page 4, line 19).

In one exemplary embodiment, a duration of said program task is defined by said one or more conditions associated with a state of said software system (page 7, line 4, to page 9, line 7).

5 In one exemplary embodiment, said one or more conditions includes an entry or exit of at least one specified method (page 5, line 1, to page 9, line 7).

In one exemplary embodiment, one or more conditions includes a creation or deletion of at least one specified object (page 5, line 1, to page 9, line 7).

10 In one exemplary embodiment, said one or more conditions includes a passing of at least one specified object or scalar value as an argument, return value or field value (page 5, line 1, to page 9, line 7).

In one exemplary embodiment, said one or more conditions includes at least one specified sequence of method invocations (page 5, line 1, to page 9, line 7).

15 In one exemplary embodiment, said one or more conditions includes at least one specified resource exceeding at least one specified threshold (page 5, line 1, to page 9, line 7).

20 In one exemplary embodiment, a method for tracing details associated with a program task executing in a software system is disclosed (page 3, line 22, to page 4, line 19), comprising: monitoring said software system to identify said program task based on a specification associated with said program task, wherein said specification contains one or more conditions to initiate a trace of said program task (page 4, line 20, to page 9, line 7); and providing trace details associated with said program task (page 3, line 22, to page 4, line 19).

25 In one exemplary embodiment, a system (110, 120) for analyzing behavior of a software system is disclosed (page 3, line 22, to page 4, line 19), comprising: a memory that stores computer-readable code; and a processor operatively coupled to said memory, said processor configured to implement said computer-readable code, said computer-readable code configured to: collect details associated with a program task associated with said software system based on a specification associated with said program task, wherein said specification contains one or more conditions to initiate a trace of said program task (page 4, line 20, to page 9, line 7); and provide said collected

details for analysis (page 3, line 22, to page 4, line 19).

In one exemplary embodiment, a system for tracing details associated with a program task executing in a software system is disclosed (page 3, line 22, to page 4, line 19), comprising: a memory that stores computer-readable code; and a processor operatively coupled to said memory, said processor configured to implement said computer-readable code, said computer-readable code configured to: monitor said software system to identify said program task based on a specification associated with said program task, wherein said specification contains one or more conditions to initiate a trace of said program task (page 4, line 20, to page 9, line 7); and provide trace details associated with said program task (page 3, line 22, to page 4, line 19).

In one exemplary embodiment, an article of manufacture for analyzing behavior of a software system is disclosed (page 3, line 22, to page 4, line 19), comprising: a computer readable medium having computer readable code means embodied thereon, said computer readable program code means comprising: a step to collect details associated with a program task associated with said software system based on a specification associated with said program task, wherein said specification contains one or more conditions to initiate a trace of said program task (page 4, line 20, to page 9, line 7); and a step to provide said collected details for analysis (page 3, line 22, to page 4, line 19).

In one exemplary embodiment, an article of manufacture for tracing details associated with a program task executing in a software system is disclosed (page 3, line 22, to page 4, line 19), comprising: a computer readable medium having computer readable code means embodied thereon, said computer readable program code means comprising: a step to monitor said software system to identify said program task based on a specification associated with said program task, wherein said specification contains one or more conditions to initiate a trace of said program task (page 4, line 20, to page 9, line 7); and a step to provide trace details associated with said program task (page 3, line 22, to page 4, line 19).

STATEMENT OF GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-2, 17-18, 20-23, and 28-30 are rejected under 35 U.S.C. §101 because the claimed invention is directed to non-statutory subject matter. Claims 1-35 are rejected under 35 U.S.C. §102(b) as being anticipated by Laffra et al.
5

ARGUMENT

Section 101 Rejections

Claims 1-2, 17-18, 20-23, and 28-30 remain rejected under 35 U.S.C. §101 because the claimed invention is directed to non-statutory subject matter. In particular, the Examiner asserts that the claims fail to technologically embody the invention in a tangible medium (i.e., a computer readable medium), and consequently fail to produce a tangible or useful result. In addition, the Examiner asserts that the cited claims are only directed to abstract ideas (i.e., details associated with a program task) and therefore do not contain an “article” to be reduced “to a different state or thing.” The Examiner further asserts that the cited claims recite only the gathering and manipulation of abstract ideas and consequently recite no more than a mathematical algorithm.
10
15

Appellants note that the collection or trace of details for analysis is more than a data gathering method. The trace of details is not simply a listing of the original program code; it is a summary of the execution of such code and therefore constitutes a transformation of the original code. As such, it meets the standard for the transformation of subject matter “to a different state or thing” as stated in *In re Warmerdam*.

Thus, as expressly set forth in each of the amended independent claims, the claimed methods or system collect or trace details associated with a program task, and provide the collected or traced details for analysis. This transformation to a collection or trace of details for analysis provides a useful, concrete and tangible result.
20
25

Appellants submit that each of the claims 1-35 are in full compliance with 35 U.S.C. §101, and accordingly, respectfully request that the rejection under 35 U.S.C. §101 be withdrawn.

Independent Claims 1, 24 and 32-35

Independent claims 1, 24, and 32-35 remain rejected under 35 U.S.C. §102(b) as being anticipated by Laffra et al. Regarding claim 1, the Examiner asserts that 5 Laffra discloses collecting details associated with a program task associated with said software system (col. 1, lines 61-63) based on a specification associated with said program task (col. 5, lines 43-48; “this specification is done using the visualization script rules”), wherein said specification contains one or more conditions to initiate a trace of said program task (col. 6, lines 12-15; “if the hook notifies creation or deletion of an 10 instance, the visualization script 285 will be used to generate or remove a visual representation of the instance”)

Appellants note that, regarding the visualization script cited by the Examiner, Laffra teaches that

15 the script 285 will tell the monitoring function (FIG. 3) *how to interpret the information which is generated by the hooks 260 and 270.* The visualization script interpretation process is described in FIG. 4. (Col. 5, lines 39-42; emphasis added.)

Laffra also teaches that

20 the monitoring function receives *information generated by hooks 260 and 270, when they are executed at runtime.* The information that is gathered by the monitoring function is then visualized on a graphics display, guided by the set of rules 288, to be found in the visualization script 285. Each time a particular hook is executed, the monitoring function 300 will inspect the current display and the script. 25 The monitoring function 300 then modifies the display depending on the hook and the visualization script.

(Col. 5, line 65, to col. 6, line 7; emphasis added.)

Laffra does **not** disclose or suggest that the hooks 260 and 270 are *conditional instructions*, and thus a person of ordinary skill in the art would recognize that hooks 260 and 270 are executed whenever they are encountered. Laffra also does 30 **not** disclose or suggest that the script 285 determines or controls *when information is generated by the hooks 260 and 270*, and does **not** disclose or suggest that the script 185 determines or controls *what information is generated by the hooks 260 and 270*. Laffra therefore does not disclose or suggest utilizing one or more conditions to **initiate a trace**

of a program task. Independent claims 1, 24, and 32-35 require collecting details associated with a program task associated with said software system based on a specification associated with said program task, wherein said specification contains one or more conditions to *initiate a trace* of said program task or monitoring said software system to identify said program task based on a specification associated with said program task, wherein said specification contains one or more conditions to *initiate a trace* of said program task.

Thus, Laffra does not disclose or suggest collecting details associated with a program task associated with said software system based on a specification associated with said program task, wherein said specification contains one or more conditions to initiate a trace of said program task, as required by independent claims 1, 32, and 34, and does not disclose or suggest monitoring said software system to identify said program task based on a specification associated with said program task, wherein said specification contains one or more conditions to initiate a trace of said program task, as required by independent claims 24, 33, and 35.

Claims 2, 3, 4 and 25

Regarding claims 2 and 25, the Examiner asserts that Laffra discloses that a duration of said program task is defined (FIG. 3: step 330) by said one or more conditions associated with a state of said software system (col. 2, lines 8-10). Regarding claim 3, the Examiner asserts that Laffra discloses said one or more conditions includes an entry or exit of at least one specified method (col. 2, lines 8-11). Regarding claim 4, the Examiner asserts that Laffra discloses said one or more conditions includes a creation or deletion of at least one specified object (col. 2, lines 8-10).

Appellants note that, in the text cited by the Examiner, Laffra discloses that

the method hooks are novelly added by an automatic technique. When a method hook is run, it can indicate the occurrence of: an object instance being created, an object instance being destroyed, a method being entered or a method being exited. In the event of one of these occurrences, the method hooks initiate execution of a monitoring function that uses the graphical information and a visualization script with one or more rules to update a visualization shown on the graphical interface.

(Col. 2, lines 7-15.)

Appellants could find no disclosure or suggestion by Laffra that a *duration of a program task* is defined by said one or more conditions associated with a state of the software system, that said one or more conditions includes an entry or exit of at least one specified method, or that said one or more conditions includes a creation or deletion of at least one specified object. Claims 2 and 25 require wherein a duration of said program task is defined by said one or more conditions associated with a state of said software system. Claim 3 requires wherein said one or more conditions includes an entry or exit of at least one specified method. Claim 4 requires wherein said one or more conditions includes a creation or deletion of at least one specified object.

Thus, Laffra does not disclose or suggest wherein a duration of said program task is defined by said one or more conditions associated with a state of said software system, as required by claims 2 and 25, does not disclose or suggest wherein said one or more conditions includes an entry or exit of at least one specified method, as required by claim 3, and does not disclose or suggest wherein said one or more conditions includes a creation or deletion of at least one specified object, as required by claim 4.

Claims 6 and 8

Regarding claim 6, the Examiner asserts that Laffra discloses said one or more conditions includes a passing of at least one specified object or scalar value as an argument, return value or field value (col. 9, lines 16-20 and 6-10) Regarding claim 8, the Examiner asserts that Laffra discloses said one or more conditions includes at least one specified resource exceeding at least one specified threshold (col. 9, lines 6-10).

In the text cited by the Examiner, Laffra teaches that

syntactical elements 440 allow the specification of one or more method triggers. A method trigger refers to a class name, and a method name, and defines what to do when hooks 260 or 270, executed at runtime (310), match this trigger. Hooks 260 and 270 mention the class name and the method name. When the class name of the hook matches the class name of the trigger, and the method name of the hook matches the method name of the trigger, the trigger matches the hook, and the corresponding action is executed. *The action that is executed comprises assigning new values to global variables or local variables.* In some preferred embodiments actions may also be the generation of a given tone on a audio device, or the activation of a sleep statement--resulting in a

delay of the visualization, or any aspect influencing the currently running visualization.

(Col. 9, lines 16-30; emphasis added.)

5 Appellants, however, could find no disclosure or suggestion by Laffra that “assigning new values to global variables or local variables” is a *condition that defines a duration of said program task*, or that conditions include at least one specified resource exceeding at least one specified threshold. Claim 6 requires wherein said one or more conditions includes a passing of at least one specified object or scalar value as an argument, return 10 value or field value that defines a duration of said program task. Claim 8 requires wherein said one or more conditions includes at least one specified resource exceeding at least one specified threshold.

Thus, Laffra does not disclose or suggest wherein said one or more conditions includes a passing of at least one specified object or scalar value as an argument, return value or field value that defines a duration of said program task, as required by claim 6, and does not disclose or suggest wherein said one or more conditions includes at least one specified resource exceeding at least one specified threshold, as required by claim 8.

Claim 7

20 Regarding claim 7, the Examiner asserts that Laffra discloses said one or more conditions includes at least one specified sequence of method invocations (col. 7, lines 41-43).

Appellants note that, in the text cited by the Examiner, Laffra teaches that

25 method triggers can be used to, for instance, *count the number of times a given method is executed*, and to update the display when a certain threshold has been reached. Furthermore, method triggers can be used, for example, to change the display when a given method is entered, for instance by coloring a visual item red, and to reset the display when the method is left again, for instance by coloring the visual item 30 back to green.

(Col. 7, lines 40-48; emphasis added.)

Appellants, however, could find no disclosure or suggestion by Laffra that a *specified sequence* of method invocations is a *condition that defines a duration of said program task*. Claim 7 requires wherein said one or more conditions includes at least one

specified sequence of method invocations.

Thus, Laffra does not disclose or suggest wherein said one or more conditions includes at least one specified sequence of method invocations, as required by claim 7.

5

Conclusion

The rejections of the cited claims under section 102 in view of Laffra et al. are therefore believed to be improper and should be withdrawn. The remaining rejected dependent claims are believed allowable for at least the reasons identified above with
10 respect to the independent claims.

The attention of the Examiner and the Appeal Board to this matter is appreciated.

Respectfully,

15

Kevin M. Mason

20

Date: November 13, 2006

Kevin M. Mason
Attorney for Applicant(s)
Reg. No. 36,597
Ryan, Mason & Lewis, LLP
1300 Post Road, Suite 205
Fairfield, CT 06824
(203) 255-6560

APPENDIX

1. A method for analyzing behavior of a software system, comprising:
collecting details associated with a program task associated with said software system
5 based on a specification associated with said program task, wherein said specification contains one or more conditions to initiate a trace of said program task; and providing said collected details for analysis.

2. The method of claim 1, wherein a duration of said program task is defined
10 by said one or more conditions associated with a state of said software system.

3. The method of claim 2, wherein said one or more conditions includes an entry or exit of at least one specified method.

- 15 4. The method of claim 2, wherein said one or more conditions includes a creation or deletion of at least one specified object.

5. The method of claim 2, wherein said one or more conditions includes an invocation of at least one specified object.

- 20 6. The method of claim 2, wherein said one or more conditions includes a passing of at least one specified object or scalar value as an argument, return value or field value.

- 25 7. The method of claim 2, wherein said one or more conditions includes at least one specified sequence of method invocations.

8. The method of claim 2, wherein said one or more conditions includes at least one specified resource exceeding at least one specified threshold.

30

9. The method of claim 1, wherein said collected details include an existence or sequence of specified method invocations.

10. The method of claim 1, wherein said collected details include an existence or sequence of specified object creations and deletions.

5

11. The method of claim 1, wherein said collected details include an existence or sequence of specified class loading and unloading.

10 12. The method of claim 1, wherein said collected details include values of specified arguments to invocations of specified methods.

13. The method of claim 1, wherein said collected details include values of specified return values from invocations of specified methods.

15

14. The method of claim 1, wherein said collected details include values of specified field values for invoked objects or field values for passed arguments.

15. The method of claim 1, further comprising the step of collecting said details for at least one specified number of task instances

20

16. The method of claim 1, further comprising the step of collecting said details for at least one specified number of threads.

25 17. The method of claim 1, further comprising the step of dynamically modifying said specification associated with said program task associated with said analysis in an iterative process.

18. The method of claim 1, further comprising the step of dynamically modifying said specification to identify which details to collect in an iterative process.

30

19. The method of claim 1, further comprising the step of connecting to a running version of said software system.

20. The method of claim 1, further comprising the step of visually analyzing
5 said collected details.

21. The method of claim 1, further comprising the step of visually analyzing said collected details for a plurality of instances of said program task.

10 22. The method of claim 1, further comprising the step of quantitatively analyzing said collected details.

23. The method of claim 1, further comprising the step of quantitatively analyzing said collected details for a plurality of instances of said program task.

15 24. A method for tracing details associated with a program task executing in a software system, comprising:

monitoring said software system to identify said program task based on a specification associated with said program task, wherein said specification contains one
20 or more conditions to initiate a trace of said program task; and

providing trace details associated with said program task.

25. The method of claim 24, wherein a duration of said program task is defined by said one or more conditions associated with a state of said software system.

26. The method of claim 25, wherein said one or more conditions is selected from the group consisting essentially of (i) an entry or exit of at least one specified method, (ii) a creation or deletion of at least one specified object, (iii) an invocation of at least one specified object, (iv) a passing of at least one specified object or scalar value as an argument, return value or field value, (v) at least one specified sequence of method

invocations, and (vi) at least one specified resource exceeding at least one specified threshold.

27. The method of claim 24, wherein said collected details include at least one
5 of the following: (i) an existence or sequence of specified method invocations, (ii) an
existence or sequence of specified object creations and deletions, (iii) an existence or
sequence of specified class loading and unloading, (iv) values of specified arguments to
invocations of specified methods; (v) values of specified return values from invocations
of specified methods, and (v) values of specified field values for invoked objects or field
10 values for passed arguments.

28. The method of claim 24, further comprising the step of collecting said
details for at least one of at least one specified number of task instances and at least one
specified number of threads

15
29. The method of claim 24, further comprising the step of dynamically
modifying a specification associated with said program task associated with said analysis
in an iterative process.

20 30. The method of claim 24, further comprising the step of dynamically
modifying said specification to identify which details to collect in an iterative process.

31. The method of claim 24, further comprising the step of connecting to a
running version of said software system.

25
32. A system for analyzing behavior of a software system, comprising:
a memory that stores computer-readable code; and
a processor operatively coupled to said memory, said processor configured
to implement said computer-readable code, said computer-readable code configured to:
30 collect details associated with a program task associated with said

software system based on a specification associated with said program task, wherein said specification contains one or more conditions to initiate a trace of said program task; and provide said collected details for analysis.

5 33. A system for tracing details associated with a program task executing in a software system, comprising:

a memory that stores computer-readable code; and

a processor operatively coupled to said memory, said processor configured to implement said computer-readable code, said computer-readable code configured to:

10 monitor said software system to identify said program task based on a specification associated with said program task, wherein said specification contains one or more conditions to initiate a trace of said program task; and

provide trace details associated with said program task

15 34. An article of manufacture for analyzing behavior of a software system, comprising:

a computer readable medium having computer readable code means embodied thereon, said computer readable program code means comprising:

20 a step to collect details associated with a program task associated with said software system based on a specification associated with said program task, wherein said specification contains one or more conditions to initiate a trace of said program task; and a step to provide said collected details for analysis.

25 35. An article of manufacture for tracing details associated with a program task executing in a software system, comprising:

a computer readable medium having computer readable code means embodied thereon, said computer readable program code means comprising:

30 a step to monitor said software system to identify said program task based on a specification associated with said program task, wherein said specification contains one or more conditions to initiate a trace of said program task; and

a step to provide trace details associated with said program task.

EVIDENCE APPENDIX

There is no evidence submitted pursuant to § 1.130, 1.131, or 1.132 or entered by the Examiner and relied upon by appellant.

RELATED PROCEEDINGS APPENDIX

There are no known decisions rendered by a court or the Board in any proceeding identified pursuant to paragraph (c)(1)(ii) of 37 CFR 41.37.